Kompatibilitätsanalyse dynamischen Verhaltens von integrierten Automobil-Steuergeräten

Matthias Glockner[a], Wolfram Hardt[b], Maximilian Fuchs[c], Michael Macht[d]
[a] BMW AG, Forschungs- und Innovationszentrum, 80788 München, Germany, email: Matthias.Glockner@bmw.de, +49-89-382-55577
[b] TU-Chemnitz, Technische Informatik, Straße der Nationen 62, 09111 Chemnitz, email: hardt@cs.tu-chemnitz.de, +49-371-531-1467
[c] BMW AG, Forschungs- und Innovationszentrum, 80788 München, Germany, email: Maximilian.Fuchs@bmw.de, +49-89-382-21189
[d] BMW AG, Forschungs- und Innovationszentrum, 80788 München, Germany, email: Michael.Macht@bmw.de, +49-89-382-40888

1 Motivation

In Premium-Fahrzeugen werden derzeit ca. 60-70 Steuergeräte (SG) verbaut. Steuergeräte sind „eingebettete Systeme“ und stellen die zentralen Rechen- und Steuereinheiten für die elektrischen und elektronischen Systeme im Fahrzeug dar. Mit Hilfe der Steuergeräte, die über Bus-Systeme (z.B. CAN-Bus) miteinander vernetzt sind, können komplexe Funktionen realisiert werden. Das Motor-Steuergerät ist eines der bekanntesten.

Der steigende Innovationsgrad im Elektrik- / Elektronik-Umfeld zwingt die Autohersteller, in immer kürzeren Zeitabständen, neue Steuergeräte mit höherer Rechenleistung und neuen Funktionen zu entwickeln. Aufgrund der steigenden Anzahl von Steuergeräten, spielt deren Rückwärtskompatibilität eine immer wichtigere Rolle. So hat Rückwärtskompatibilität beispielsweise folgende Vorteile:

- Reduzierung von Entwicklungskosten, durch direkte Übernahme von neuen Steuergeräten in weitere Fahrzeugmodelle und
- Wegfall der Bevorratung von alten Steuergeräten als Ersatzteile und damit Senkung der Gewährleistungskosten.

Das Forschungsprojekt[1] „Compatibility Analysis for Electronic Control Units“ (CompA) beschäftigt sich mit Methoden zur Analyse von Rückwärtskompatibilität. Der Fokus liegt auf der Frage: „Kann ein neu entwickeltes Steuergerät (SG_2) ein Vorgänger-Steuergerät (SG_1), im selben Fahrzeug- oder in anderen Fahrzeugmodellen, ersetzen d.h. ist SG_2 rückwärtskompatibel zu SG_1 (Abbildung 1)?“

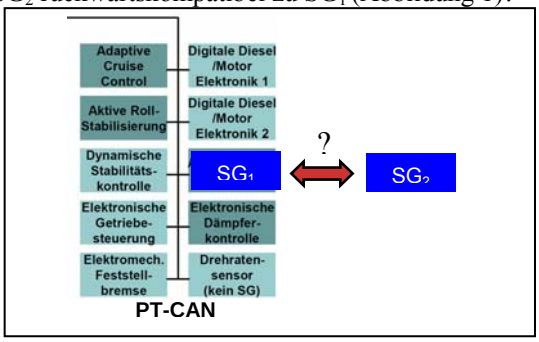

Abbildung 1 Ausschnitt des PowerTrain-CAN und Betrachtung der Rückwärtskompatibilität von SG_2

In diesem Paper wird ein Teilgebiet des CompA-Ansatzes vorgestellt. Der Schwerpunkt ist hierbei die Analyse von Rückwärtskompatibilität bzgl. des dynamischen Verhaltens von Steuergeräten.

[1] Eine Zusammenarbeit der BMW Group und der TU-Chemnitz (Professur Technische Informatik)

2 Forschungsprojekt „CompA“

Automobilhersteller spezifizieren Steuergeräte durch „Lastenhefte“, wobei die Steuergeräte hierbei als „Blackboxes“ betrachtet werden. Aus diesem Grund fokussiert das Forschungsprojekt „CompA“ auf einen durchgängigen und gesamthaften Ansatz, der es ermöglicht, Rückwärts-Kompatibilität von Steuergeräten auf Basis von Spezifikationen (Lastenheften) zu analysieren.

Der Ansatz baut auf 3 Ebenen auf:

1. *Formalisierung:* Definition eines formalen XML-Schemas, anhand dessen Steuergeräte hinreichend beschrieben werden können.
2. *Instanziierung:* Bereitstellung von Hilfsmitteln (Graphikeditor) zur teilweise automatisierten (XML-) Beschreibung von Steuergeräten
3. *Vergleichsmethoden:* Definition von Methoden zur Analyse von Rückwärts-Kompatibilität auf Basis von Spezifikationen.

3 Kompatibilitäts-Betrachtungen von dynamischen Verhalten

Steuergeräte werden sowohl durch statische Eigenschaften (Größe, Signaleingänge, etc.) als auch durch dynamische Eigenschaften (funktionales/dynamisches Verhalten) beschrieben. Dieses Paper stellt ein spezifisches Teilgebiet des CompA-Ansatzes vor, die „Kompatibilitätsanalyse von dynamischen Verhalten von Steuergeräten anhand von Spezifikationen“.

Hierbei werden folgende Punkte genauer betrachtet:

1. *MSC-Beschreibung:* Zur Spezifikation des funktionalen/dynamischen Verhaltens am Interface (Schnittstelle) von Steuergeräten werden Message Sequence Charts (MSC) verwendet.
2. *Kompatibilität:* Definition von Kompatibilität von MSC-Beschreibungen.
3. *MSC-Vergleichsmethode:* Vergleich von MSCs durch Transformation in Automaten und Analyse bzgl. Kompatibilität.

3.1 MSC-Beschreibung

Message Sequence Charts (MSC) wurden ursprünglich zur Beschreibung von Kommunikationsverhalten entworfen [1], können aber auch zur Beschreibung von Schnittstellenverhalten von Steuergeräten verwendet werden.

Ein MSC ist ein gerichteter, nicht zyklischer Graph, der formal wie folgt definiert ist [2]:

- *P ist endliche Menge von Prozessen.*
- *S ist eine endliche Menge von Sendeereignissen und R eine endliche Menge von Empfangsereignissen, mit* $S \cap R = \varnothing$. *Die Menge* $S \cup R$ *wird mit E bezeichnet.*
- $L : E \to P$ *ordnet jedem Ereignis e einen eindeutigen Prozess* $L(e) \in P$ *zu. Die Menge der Ereignisse, die einem Prozess p zugeordnet werden, sei mit* E_p *bezeichnet.*
- $c : S \to R$ *ist eine bijektive Abbildung, die jedem Sendeereignis s genau ein Empfangsereignis* $c(s)$ *und jedem Empfangsereignis r genau ein Sendeereignis* $c^{-1}(r)$ *zuordnet.*
- $<_p$ *ist eine lokale Totalordnung, die für alle* $p \in P$ *die Ereignisse* E_p *linear ordnet. Die Ordnung korrespondiert mit der im MSC abgebildeten Ordnung.*

Der Vorteil von MSCs ist die graphische Repräsentation einer Verhaltensbeschreibung. Jedoch kann durch ein MSC immer nur ein spezifisches Scenario eines Verhaltens beschrieben werden. Eine komplette Verhaltensbeschreibung kann nur durch ein Set von MSC's erfolgen.

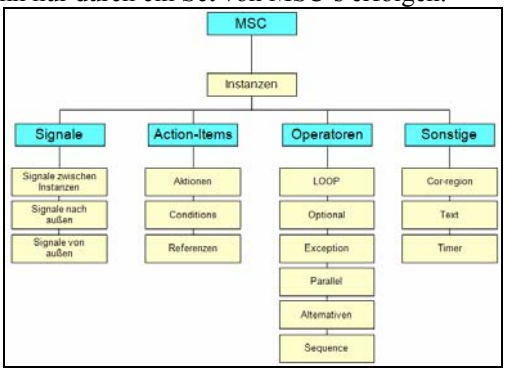

Abbildung 2 Baukasten zur Erstellung von MSCs.

Zur Beschreibung von funktionalem Verhalten, stellt die MSC-Sprache einen „Baukasten" an verschiedenen Elementen zur Verfügung (vgl. Abbildung 2). Dieser Baukasten enthält u.a. Elemente zur Beschreibung von Schleifen, parallelen Abläufen, Bedingungen, Timer, etc..[1, 3]. In Abbildung 3 ist ein einfaches Beispiel für die Mächtigkeit der MSC-Sprache dargestellt. Das Signal x_1 wird an den Prozess „Instance_1" gesendet. Anschließend folgt ein *Optional-Block*.

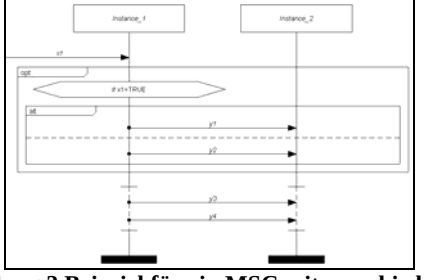

Abbildung 3 Beispiel für ein MSC mit verschiedenen Beschreibungselementen

Dies bedeutet, dieser Block kann, muss aber nicht ausgeführt werden. Wird der Block ausgeführt und ist das Signal x_1=True, so ist der *Condition-Block* „gültig" und der nachfolgende *Alternativ-Block* wird ausgeführt d.h. es wird entweder das Signal y_1 oder y_2 gesendet. Unabhängig von der Ausführung des *Optional-Blocks* werden am Ende die Signale y_3 und y_4 gesendet. Da die beiden Signale in einer *Co-Region* abgebildet sind, ist die Reihenfolge, in der die Signale y_3 und y_4 gesendet werden, egal.
Zur Erstellung von MSCs gibt es bereits gute Best - Practices-Dokumentationen [3].

Herausforderung

Aufgrund der Mächtigkeit der MSC-Sprache kann gleiches Verhalten unterschiedlich beschrieben werden. In Abbildung 4 sind zwei MSCs dargestellt, die einen unterschiedlichen Aufbau haben. Die Ereignissequenzen (Abfolge von möglichen Input/Output-Sequenzen) sind jedoch identisch. Damit beschreiben die beiden MSCs ein äquivalentes dynamisches Verhalten.

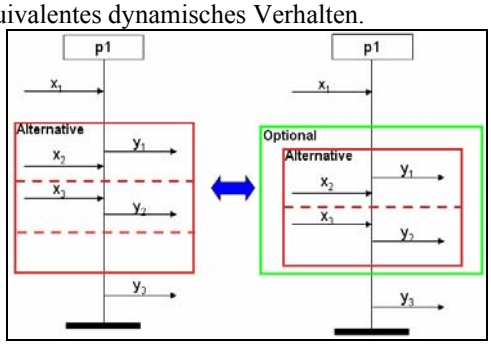

Abbildung 4 MSC mit unterschiedlicher Darstellung, aber äquivalentem Verhalten.

Da also gleiches Verhalten unterschiedlich dargestellt werden kann, ist ein „einfacher" Vergleich von MSCs nicht ausreichend. In Kapitel 5 wird hierfür eine Lösung vorgestellt.

3.2 Kompatibilität

Kompatibilität zweier MSC sei wie folgt definiert:
Satz (1): „*Zwei MSCs sind genau dann zueinander rückwärtskompatibel, wenn Ihre Ereignissequenzen äquivalent sind und die Ereignisse (Signale) zueinander kompatibel sind.*"
Formal geschrieben:

$$(MSC_1 \Leftrightarrow_c MSC_2) \Leftrightarrow$$
$$(<_{p_i} \Leftrightarrow <_{p_k} \wedge E_{p_i} \Leftrightarrow_c E_{p_k} \,|\, p_i \in MSC_1; p_k \in MSC_2)$$

E_{p_i}, E_{p_k} : Ereignisse eines Prozesses

$<_{p_i}, <_{p_k}$: geordnete Reihenfolge von E_{p_i}, E_{p_k}

$\Leftrightarrow_c$: kompatibel

4 Stand der Technik

[4] beschreibt eine Methode, mit der mehrere MSCs, die jeweils ein Scenario beschreiben, zu einer gesamthaften Verhaltensbeschreibung zusammengeführt werden können. Die gesamthafte Verhaltensbeschreibung erfolgt mittels ω-Automaten auf. Obwohl die Ansätze

unterschiedliche Ziele verfolgen, werden im CompA-Ansatz einzelne Ideen/Ansätze als Synergien genutzt.

5 MSC-Vergleichsmethode

Um unterschiedliche MSC gegeneinander vergleichen zu können, müssten diese auf eine normierte Darstellung gebracht werden. Für diese Problematik gibt es in der Automatentheorie unter den Stichpunkten Determinierung, Reduktion, etc. bereits unterschiedliche Lösungsansätze [5]. Da MSCs analog zu Automaten ein Ein-/Ausgangsverhalten beschreiben, wird im CompA-Ansatz eine Transformation von MSCs in Automaten vorgeschlagen (vgl. Abbildung 3).
Dadurch kann das Problem der unterschiedlichen Darstellungsmöglichkeiten von MSCs elegant gelöst werden. Bzgl. der Transformation von MSCs in Automaten wird eine Erweiterung des Ansatzes von [4] vorgestellt.

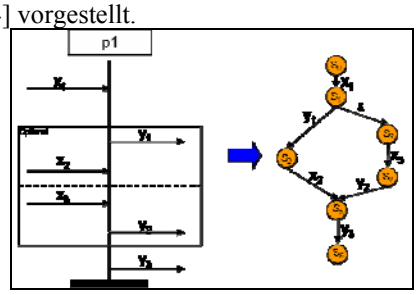

Abbildung 5 Transformation eines MSC's in einen Automaten

Die im CompA-Ansatz vorgeschlagene Methode gliedert sich in folgende Teilschritte (vgl. Abbildung 4):

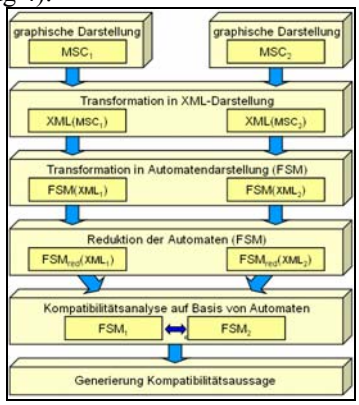

Abbildung 6 Schritte der MSC-Vergleichsmethode

1. *Einlesen von MSCs:* MSC werden graphisch erstellt und als ASCII-File abgespeichert.
2. *Transformation nach XML:* Die ASCII-Files werden mittels Parser (und XML-Schema) in eine XML-Darstellung transformiert
3. *Automaten-Transformation:* Die XML-MSC werden in Automaten transformiert.
4. *Reduktion der Automaten:* Reduktion der Automaten auf die „kleinste" mögliche Darstellung (≙ Normierung)
5. *Kompatibilitätsanalyse:* Analyse, der Automaten bzgl. Kompatibilität

5.1 Einlesen von MSCs

MSCs können mit unterschiedlichen am Markt befindlichen Tools erstellt werden. In unserem Fall wurde ein von ESG entwickelter MSC-Editor eingesetzt. Vorteil dieser Tools ist, dass die Syntax, mit denen ein MSC beschrieben werden kann, bereits vorgegeben ist. Die MSCs werden als ASCII-File abgespeichert (vgl. Abbildung 7).

```
msc Beispiel_Maechtigkeit;
gate out USERDEF.x1,1 to Instance_1;
Instance_1: instance /*FBlockName: '', InstanceId:
'', InstanceDevice: '', MostCatalogAlias:
'USERDEF', ExtensionCatalogAlias: ''*/;
Instance_2: instance /*FBlockName: '', InstanceId:
'', InstanceDevice: '', MostCatalogAlias:
'USERDEF', ExtensionCatalogAlias: ''*/;
Instance_1: label L0; in USERDEF.x1,1 from env;
all: opt begin:
```

Abbildung 7 Ausschnitt aus ASCII-File des in Abbildung 3 beschriebenen MSCs

5.2 Transformation nach XML

Die ASCII-Files werden mit Hilfe eines Parsers und eines MSC-Metamodells in eine XML-Beschreibung transformiert [6]. Dies hat zum Vorteil, dass für MSCs, die mit anderen Editoren erstellt werden, lediglich der Parser angepasst werden muss und so alle in XML vorliegenden MSC-Beschreibungen syntaktisch „gleich" sind.

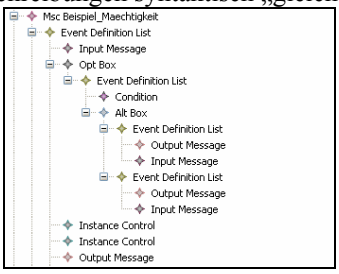

Abbildung 8 XML/EMF-Instanz des MSC aus Abbildung 3

Das im Rahmen von CompA verwendete MSC-Metamodell und der zugehörige Parser wurde bei der BMW Group auf Basis von EMF (Eclipse Modeling Framwork) entworfen. In Abbildung 8 ist die XML/EMF-Instanz [7] des Beispiels aus Abbildung 3 dargestellt. Man kann sehr gut die Struktur des MSC erkennen. Die XML/EMF-Instanz dient als Ausgangsbasis für die Automatentransformation.

5.3 Automaten-Transformation

Die Transformation von MSCs in Automaten gliedert sich in drei Bereiche:

1. *Dekomposition von MSCs*
2. *Transformation der Prozesse p_i in Basis-Automaten*
   a. Zuweisung von Zuständen
   b. Zuweisung von Transitionen und ε-Transitionen
3. *Transformation spezieller MSC-Beschreibungselemente* (Alternative-Block, ...)

5.3.1 Dekomposition von MSCs

Jeder einzelne Prozess eines MSC beschreibt analog zu Automaten ein Ein-/Ausgangsverhalten. Zur Transformation in Automaten werden MSCs daher in die einzelnen Prozesse dekomponiert (vgl. Abbildung 9).

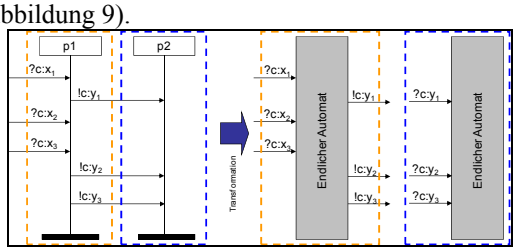

Abbildung 9 Dekomposition von MSC

Satz (2): *Ein MSC ist die Aggregation aus* $n \in \mathrm{N}$ *Einzel-Prozessen* (p_i) *mit einer eindeutigen Zuordnung der Ereignisse zu den Prozessen:*

$$MSC_x = \sum_{i=0}^{n} p_{i_x} \text{ mit } L : E \to P$$

In Verbindung mit Satz (1) kann die Kompatibilitäts-Definition wie folgt erweitert werden:

Satz (3): *Zwei MSCs sind genau dann kompatibel, wenn Ihre Einzelprozesse zueinander kompatibel sind und die Nachbarschaftsbeziehungen erhalten bleiben.*

$$\left(MSC_1 \Leftrightarrow_c MSC_2\right) \Leftrightarrow \left(\sum_{i=0}^{n} p_i \Leftrightarrow_c \sum_{k=0}^{n} p_k \,\middle|\, p_i \in MSC_1; p_k \in MSC_2\right)$$

mit $L : E \to P$

Zur Sicherstellung, dass bei einer Transformation die Nachbarschaftsbeziehungen erhalten bleiben, wird folgende Nomenklatur eingeführt:

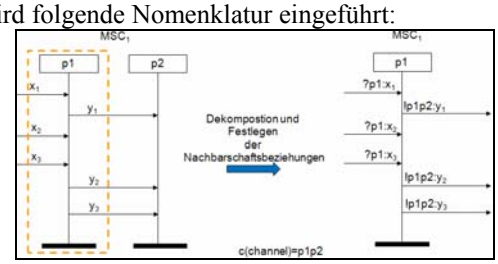

Abbildung 10 Dekomposition und Einführung einer Nomenklatur

1) Die Symbole (?,!) kennzeichnen analog zu [4] das Senden(!)/Empfangen(?).
2) Die Zuordnung der Ereignisse zu den jeweiligen Prozessen wird den Ereignisnamen vorangestellt und durch einen Doppelpunkt getrennt (vgl. Abbildung 10).

Beispiel: Wird ein Signal x_1 von Prozess p_1 zu Prozess p_2 gesendet, so sei dies künftig wie folgt dargestellt: $!p_1p_2{:}x_1$.

5.3.2 Transformation der Prozesse p_i in Basis-Automaten

Die dekomponierten MSC-Prozesse p_i werden einzeln in endliche Automaten überführt. Basis-Automaten bestehen aus Zustände und Übergangsfunktionen. Die Übergangsfunktionen beschreiben die Bedingung, über die ein Zustand in einen anderen Zustand übergehen kann. Bei der Transformation können drei verschiedene Automaten-Typen auftreten:

- *Nicht-deterministische endliche Automaten* (NEA)
- *Nicht-deterministische endliche Automaten mit ε-Transitionen* (ε-NEA)
- *Deterministische endliche Automaten* (DEA)

Die NEA-Automaten können durch verschiedene Methoden in einen DEA überführt werden (vgl. Abschnitt 5.4).

Ein Automat ist formal wie folgt beschrieben[5, 8]

$A = (Q, \Sigma, \delta, q_0, F)$, *wobei*

- Q *eine endliche Menge von Zuständen ist.*
- Σ *eine endliche Menge von Eingabesymbolen ist.*
- q_0, *ein Element von* Q, *der Startzustand ist.*
- F, *eine Teilmenge von* Q, *die Menge der finalen (oder akzeptierenden) Zustände ist*
- $\delta : Q \times \Sigma \to Q$, *die Übergangsfunktion/Transition*

Bei der Transformation werden die Ereignisse der MSCs als Übergangsbedingungen/Transitionen δ interpretiert. Durch die zuvor eingeführte Nomenklatur geht die Information, ob es sich bei dem Ereignis um ein Ein-/Ausgangssignal handelt, nicht verloren. Zwischen den Ereignissen werden Zustände eingefügt. Der erste Zustand wird immer als *initialer* Zustand gekennzeichnet und der letzte Zustand als *finaler* Zustand (diese Kennzeichnung ist für die Reduktionsalgorithmen relevant) (vgl. Abbildung 11).

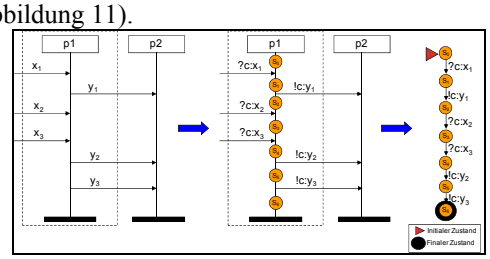

Abbildung 11 MSC-Transformation in Automaten

5.3.3 Transformation spezieller MSC-Beschreibungselemente

Einen Sonderfall der Transformation von MSCs in Automaten stellen die verschiedenen „Baukasten"-Elemente der MSCs (Loop-, Alternative-Blöcke, etc.) dar.

Um diese in Automaten zu überführen, müssen zusätzliche Zustände[2] (ε-Zustände) und zusätzliche Transitionen (ε-Transitionen) eingefügt werden. Der CompA-Ansatz stellt für alle Elemente entsprechende Transformations-Regeln bereit. Beispielhaft werden für vier Elemente diese Regeln vorgestellt.

Alternative-Block

Für einen Alternative-Block müssen bei der Transformation zusätzliche ε-Zustände eingefügt werden (vgl. Abbildung 12). Es entsteht also ein ε-NEA.

[2] ε-Zustände werden wie alle Zustände behandelt. Die Unterscheidung dient lediglich zur Verdeutlichung der Methode.

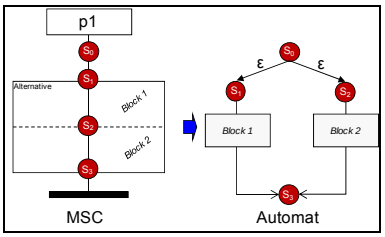

Abbildung 12 Transformation von Alternative-Block

Der Zustand S_0 dient zur Verzweigung auf die Alternative-Blöcke, die hier mit S_1 und S_2 beginnen. Die Blöcke werden auf den Zustand S_3 wieder zusammengeführt. In Abbildung 13 ist ein Beispiel dargestellt.

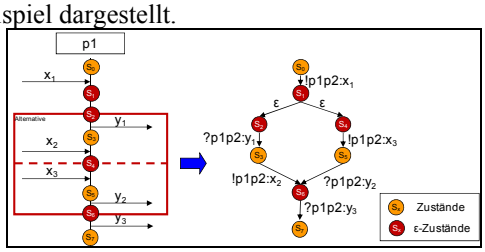

Abbildung 13 Beispiel für die Transformation eines Alternative-Blockes

Nach dem Empfang des Signals x_1 gibt es zwei Szenarien: 1) Entweder die Signale y_1 und x_2 werden gesendet bzw. empfangen oder 2) die Signale x_2 und y_2 werden empfangen bzw. gesendet. Egal, welches Szenario durchgeführt wurde, am Ende wird das Signal y_3 gesendet.

Optional-Block

Beim *Optional-Block* kann ein Block im Gegensatz zum *Alternative-Block* übersprungen werden.

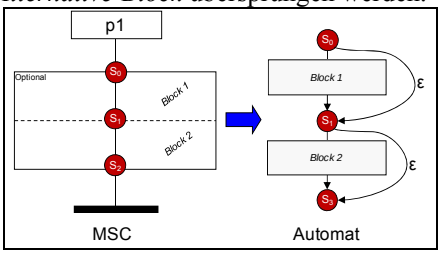

Abbildung 14 Transformation eines Optional-Blocks

Dies wird beim Automaten dadurch realisiert, dass eine direkte ε-Transition von bspw. S_0 zu S_1 eingefügt wird (vgl. Abbildung 14).

Parallel-Block

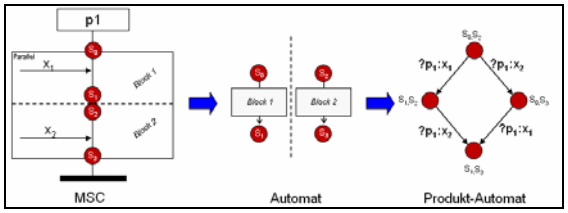

Abbildung 15 Transformation eines Parallel-Blocks

Basis-Automaten können paralleles Verhalten nicht direkt abbilden. In der Automatentheorie existieren Ansätze mit denen parallele Automatenkonstrukte in Produktautomaten [5] überführt werden können. Diese Methode kann auch hier angewendet werden. In Abbildung 15 ist dargestellt, wie ein Parallel-Block in einen Produktautomaten überführt wird. Parallele Blöcke können noch tiefer verschachtelt sein. In Abbildung 16 ist der komplexe Fall dargestellt, bei dem ein Parallel-Block einen weiteren Parallel-Block enthält. Der Parallel-Block wird in einen parallelen Automaten überführt (vgl. 1. Schritt) und so lange als paralleler Automat geführt, bis alle Elemente der Parallelen Blöcke traversiert wurden. Wird ein weiterer Parallel-Block entdeckt, wird dieser auch als paralleler Automat dargestellt (vgl. 2. Schritt). Anschließend werden die parallelen Automaten rekursiv in einen Produktautomaten überführt. So wird in unserem Beispiel erst der parallele Automat in Block 1 und anschließend die beiden übergeordneten parallelen Automaten zu einem Produktautomaten zusammengeführt.

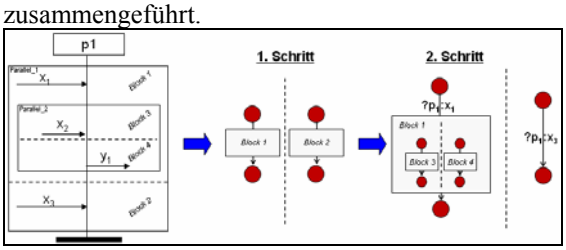

Abbildung 16 Parallel-Block durch weitere Parallel-Blöcke verschachtelt.

Timer

Mittels Timer kann in MSCs ein zeitliches Verhalten abgebildet werden. Die Timer starten und enden innerhalb einer Ereignissequenz eines MSC's. Daher können die Elemente, die einen Timer abbilden auch als Übergangsfunktionen bzw. als Transitionen abgebildet werden (vgl. Abbildung 17). Für einen Kompatibilitätsvergleich ist diese

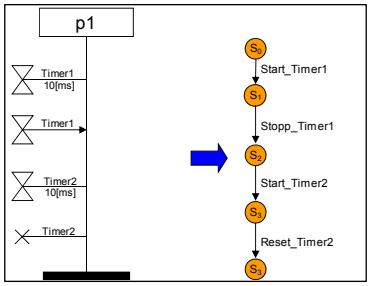

Abbildung 17 Transformation von Timer

Transformation ausreichend. Zur Unterscheidung der Timervarianten, wird den Timernamen bei der Transformation die Präfixe *Start_, Stopp_, Reset_* vorangestellt.

5.4 Reduktion der Automaten

Wie bereits angedeutet, ist eine Herausforderung, unterschiedliche Darstellungen von MSCs gegeneinander zu vergleichen. Die in Automaten überführten MSC haben einen unterschiedlichen Aufbau. Um diese „vergleichbar" gegeneinander zu machen, müssen diese auf eine normierte Darstellung gebracht werden. Für die Normierung werden in der Automatentheorie bekannte Methoden eingesetzt, die in [5] ausführlich beschrieben sind:

1. *ε-Eliminierung* Entfernen der aller ε-Transitionen.
2. *Determinierung* Umwandlung von NEA (**N**icht deterministische **E**ndliche **A**utomaten) in DEA (**D**eterministische **E**ndliche **A**utomaten)

3. *Minimierung* Reduzierung der Zustände auf die kleinste mögliche Anzahl.

5.5 Kompatibilitätsanalyse

Die Kompatibilitätsanalyse kann entsprechend der Definition (vgl. Abschnitt 3.2 und 5.3.1) in zwei Schritte durchgeführt werden:

1. *Analyse der reduzierten Automaten auf Äquivalenz*
2. *Analyse der Ereignisse auf Kompatibilität.*

Im 1. Schritt werden die Ereignissequenzen der Automaten auf Äquivalenz geprüft. Hierbei müssen sowohl die Sequenzen der Ereignisse, als auch deren Präfixe (z.B.?p_1p_2:) identisch sein. Durch die Einbeziehung der Präfixe in den Äquivalenzvergleich, wird die korrekte Beziehung der Ereignisse zu den Prozessen sichergestellt.

Im 2. Schritt werden die Signale gegeneinander auf Kompatibilität geprüft. Zwei Signale können bspw. kompatibel sein, wenn diese sich, im einfachsten Fall, nur im Namen unterscheiden. Oder ein Signal x_1 kann zu einen Signal x_2 kompatibel sein, wenn das Signal x_1 sich im Intervall von x_2 befindet.

Im Projekt „CompA“ wird die Kompatibilität aller Signale unabhängig von der MSC-Vergleichsmethode überprüft. Es wird hier also vorausgesetzt, dass alle Signale, die in den MSCs verwendet werden, auch im Rahmen des CompA-Ansatzes spezifiziert worden sind. Denn somit ist auch der Kompatibilitätsvergleich der Signale zueinander sichergestellt.

6 Beispiel

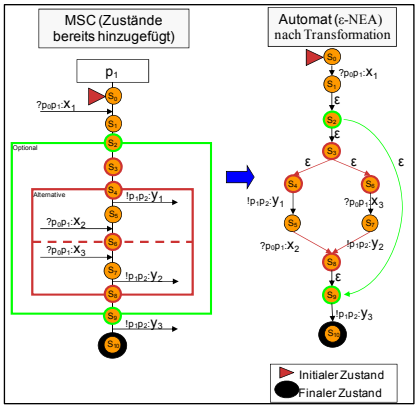

Abbildung 18 Transformation von MSC in Automat

In Abbildung 18 ist ein MSC dargestellt, das in einen Automaten überführt werden soll. Die Nomenklatur und die Zustände wurden entsprechend Kapitel 5.3.1 eingetragen und der initiale und finale Zustand gekennzeichnet. Durch die Transformation entsteht ein ε-NEA.

Durch die Eliminierung der ε-Transitionen entsteht aus dem ε-NEA ein NEA. Jeder NEA kann [5] in einen DEA transformiert werden (vgl. Abbildung 19). Haben die DEA noch nicht die minimale Anzahl an Zuständen, müssen die DEA weiter minimiert werden. In unserem Beispiel ist die minimale Anzahl an Zuständen erreicht.

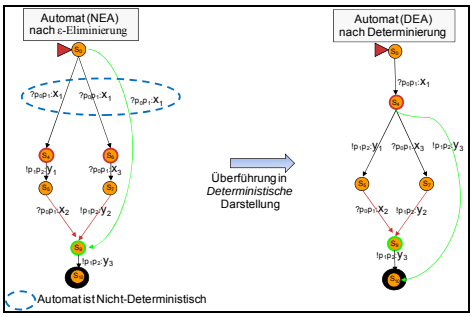

Abbildung 19 Automat nach *ε-Eliminierung* und nach *Determinierung*

Um beispielhaft zu zeigen, dass der vollständig reduzierte Automat äquivalent zu dem Ursprungsautomaten ist, wird in Abbildung 20 der reduzierte Automat rücktransformiert. Vergleicht man die Ereignissequenzen der beiden MSCs, so zeigt sich, dass diese äquivalent sind.

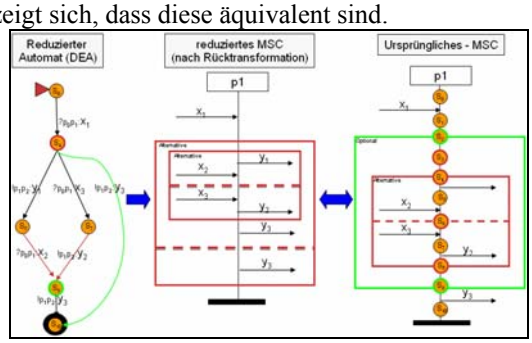

Abbildung 20 Rücktransformation des reduzierten Automaten und Vergleich mit ursprünglichem MSC

7 Zusammenfassung

Dieses Paper stellte eine Methode vor, mit der „Kompatibilität von dynamischen Verhalten von Steuergeräten anhand von Spezifikationen“ analysiert werden kann.

Die Beschreibung des dynamischen Verhaltens erfolgte durch Message Sequence Charts (MSC). Für den Vergleich der MSCs bzgl. Kompatibilität wurde gezeigt, wie MSCs in Automaten transformiert und die Automaten durch Reduktion auf eine „normierte“ Darstellung gebracht werden können. Durch einen strukturellen Vergleich der reduzierten Automaten können Aussagen bzgl. Äquivalenz und Kompatibilität der MSCs getroffen werden. Die beschriebene Methode wurde anhand eines Beispiels dargestellt.

8 Literatur

1. ITU, *Message Sequence Chart (MSC)*, O. Haugen, Editor. 2000, ITU-T.
2. Alur, R., G.J. Holzmann, and D. Peled. *An Analyzer for Message Sequence Charts*. 1996.
3. Cooperation, M., *MOST MSC Cookbook*. 2005. p. 62.
4. Krüger, I.H., *Distributed System Design with Message Sequence Charts*. 2000, Technische Universität München.
5. Hopcroft, J.M., Rajeev; Ullman, Jeffrey, *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*, ed. Auflage. 2002: Addison-Wesley.
6. Droste, M., *Effiziente Verhaltensmodellierung zur Systemintegration*. 2006, TU-München.
7. www.eclipse.org, Introducion EMF (Chapter2).
8. Schöning, U., Theoretische Informatik, kurzgefasst. 2003, Spektrum.